TRABAJO PRÁCTICO 4
Laboratorio II

Alejandro Iván Henestroza

2C

Índice

Nota preliminar

Gran parte del TP 3 fue rehecha desde 0 para adecuarse correctamente a los nuevos requerimientos.

La interfaz de usuario fue completamente repensada para brindar una mejor experiencia de usuario. Algunas entidades fueron recreadas para ajustarse a las nuevas funcionalidades.

La string de conexión a la Base de Datos es la siguiente:

```
"Data Source=.;Initial Catalog=TP4_Henestroza_Alejandro;Integrated Security=True"
```

Si se modifica el Data Source o si se le cambia el nombre a la Base de Datos, se debe modificar en el archivo DAO.cs, ubicado en el proyecto Entidades, carpeta DAO, namespace Entidades.DAO.

Introducción

Se ha desarrollado una aplicación que permite generar órdenes de producción de bicicletas y accesorios.

Hay dos tipos de bicicletas, Mountain Bike y Playera, y dos tipos de accesorios: Casco y Luz.

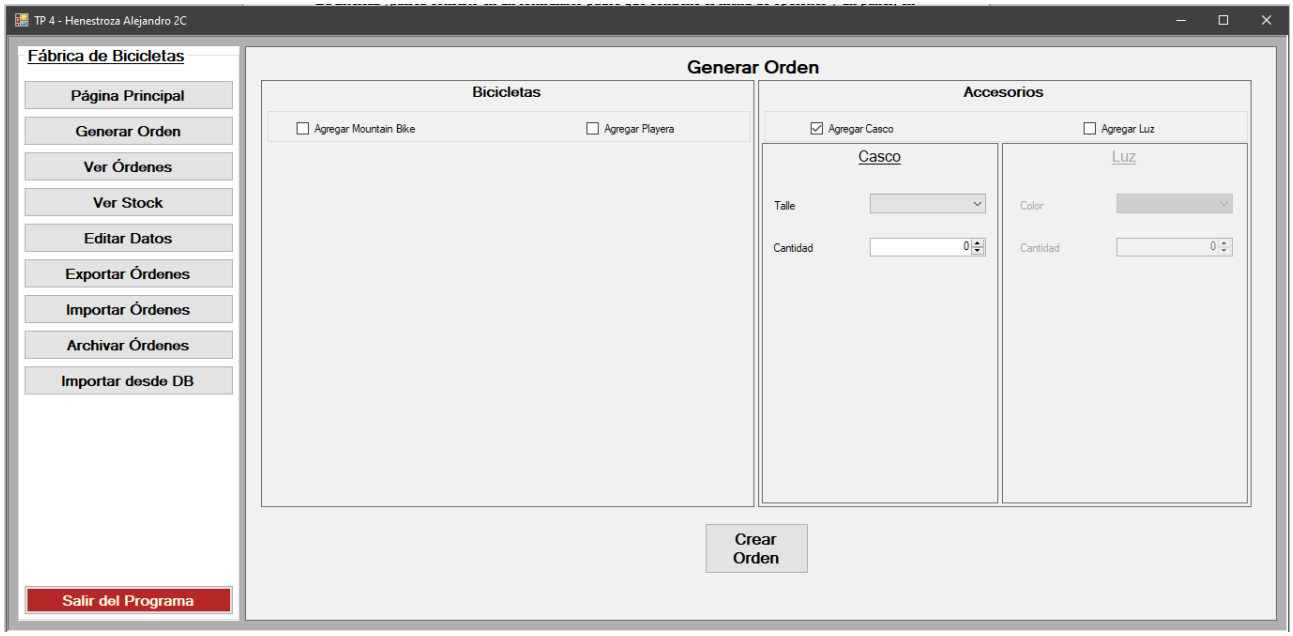
Las bicicletas se componen por dos Ruedas y un Cuadro, fabricado a partir de un Material. La aplicación sigue el stock del material (guardado en la base de datos) y permite generar nuevas órdenes si hay suficiente stock. También permite agregar o quitar stock de material.

Finalmente, la aplicación permite exportar cada orden a un archivo de texto individual, con los datos necesarios para su fabricación. También se puede exportar el listado a un archivo binario.

Interfaz Gráfica

La interfaz gráfica consiste en un formulario padre que contiene el menú de opciones y un panel, en el cual se van insertando los formularios hijos.

El botón Generar Orden abre el formulario para generar órdenes. Inicialmente, no hay controles visibles, salvo por 4 checkboxes. Al tildar un checkbox, se habilitan los controles correspondientes al componente que se desee incluir en la Orden.

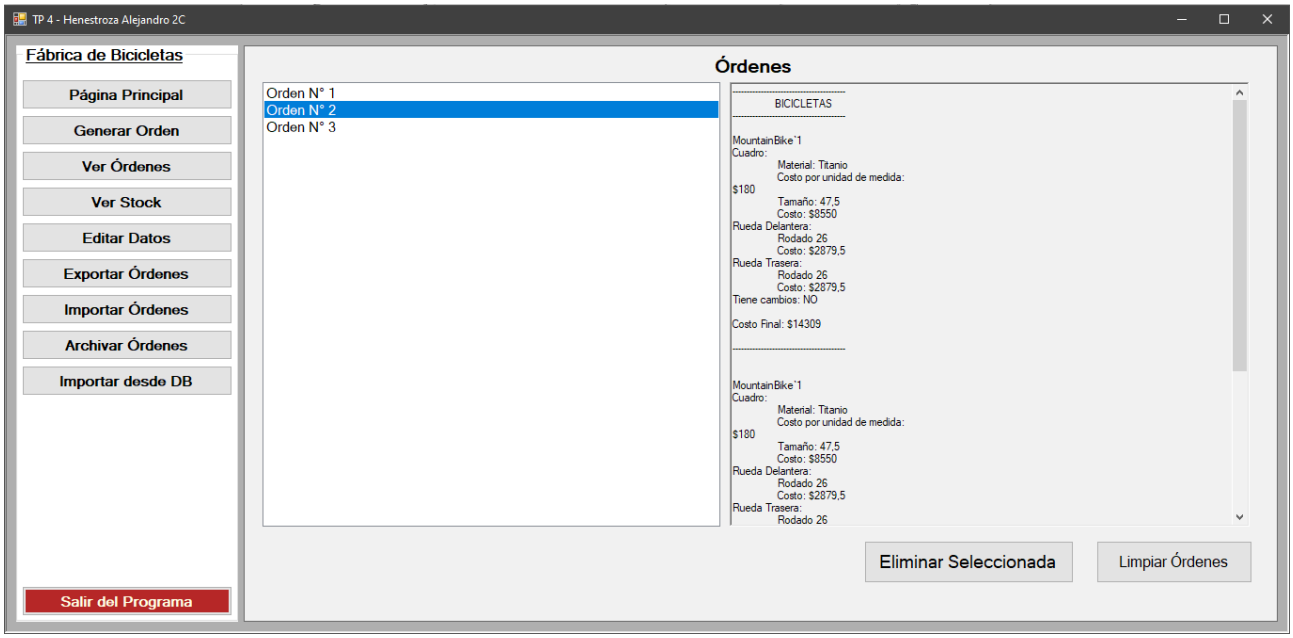


Controles de Casco habilitados, los de Luz están deshabilitados y los de Bicicletas no son visibles al no estar checkeados ninguno de los dos checkbox

Al crear la orden, se dispara el evento OrdenCreada, al cual se le suscriben los métodos manejadores desde el formulario en base a qué checkbox se haya tildado. En caso de tildar los 4, se le suscriben 4 métodos manejadores para insertar los componentes en la Base de Datos.

Antes de crear la Orden, se comprueba que haya suficiente stock de materiales para fabricar las bicicletas (en caso de que se desee incluir bicicletas).

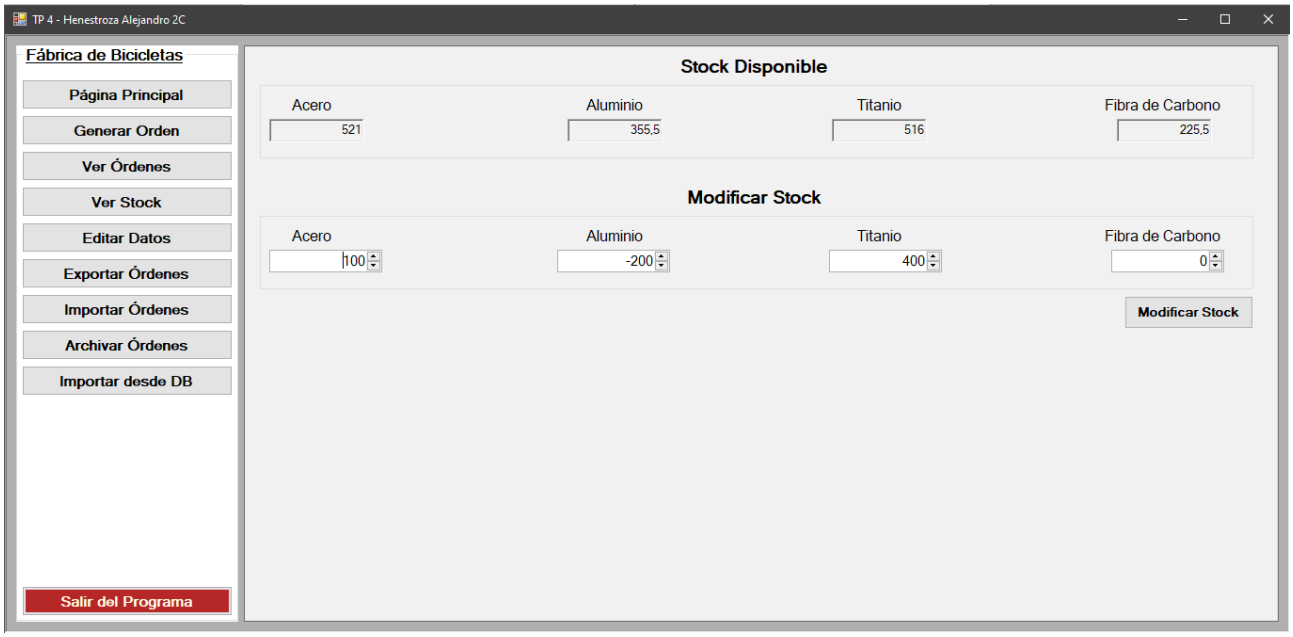
El botón Listar Órdenes muestra las órdenes creadas hasta el momento. Cada orden se puede seleccionar para ver sus datos en el panel de la derecha. Si se exporta un listado de un archivo binario o desde la Base de Datos, se listarán las órdenes del archivo o de la base.



Las órdenes listadas pueden removerse del listado y, si están guardadas en la base de datos, se eliminan.

También se puede limpiar el listado actual, sin modificar a la base o al archivo binario. O, en caso de que la orden esté en la Base de Datos, se puede eliminar del listado y de la Base.

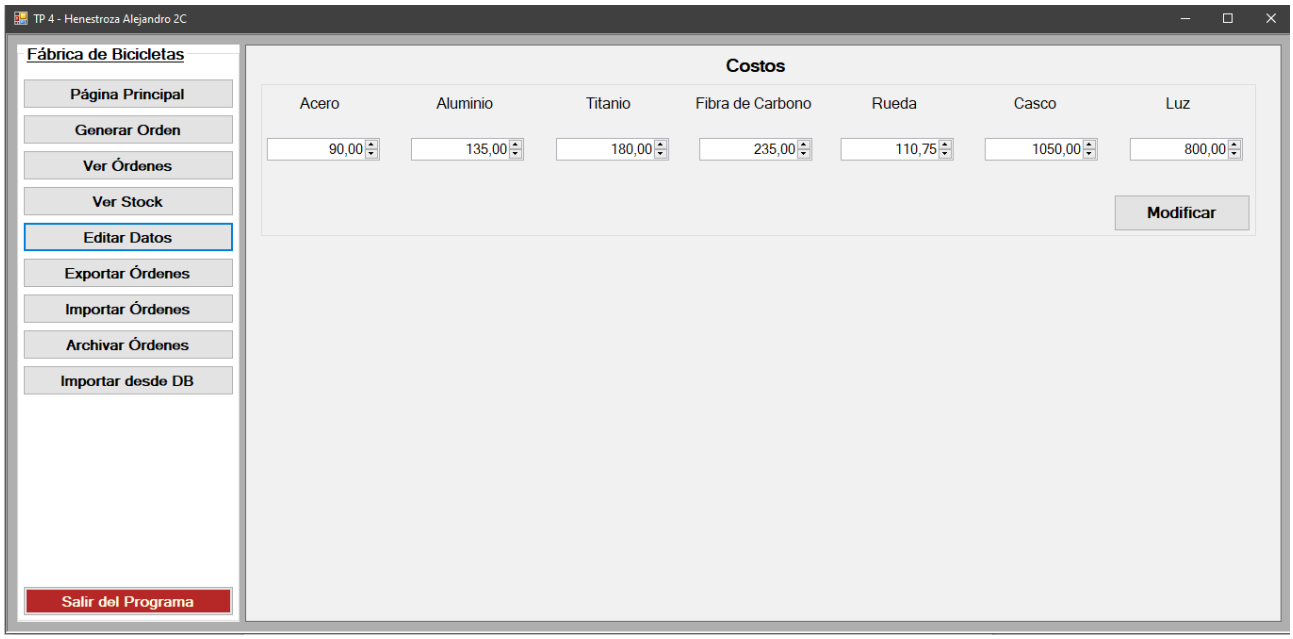
En el caso del botón Ver Stock, se abre un panel con el stock de los materiales disponibles, permitiendo también modificar el mismo.



Los incrementos son de 100 o de -100, hasta un valor máximo de 100.000 y un mínimo de -100.000. También se pueden ingresar manualmente números no múltiplos de 100

Al momento de modificar el stock, se envía una sentencia UPDATE a la base de datos para guardar los nuevos valores. En tiempo de ejecución, una clase estática almacena los valores del stock, pero los datos son persistidos en la Base de Datos.

El botón de Editar Datos permite modificar los costos base de cada material y cada accesorio (previamente los costos se encontraban hardcodeados en el código).



Los valores son cargados automáticamente al abrir el formulario, pero sólo se modifican al presionar el botón Modificar

Los costos se guardan en una tabla de la Base de Datos, así como también en una clase

estática que los sigue en tiempo de ejecución.

Exportar Órdenes e Importar Órdenes permiten exportar o importar el listado de órdenes a/desde un archivo binario. Se puede seleccionar la ubicación deseada para guardar el archivo, así como elegir que archivo se va a cargar (se controla que sea un archivo válido).

Archivar Órdenes permite elegir una carpeta donde se guardará un archivo de texto por cada orden cargada en la aplicación, que contenga los datos de la orden.

Finalmente, Importar desde DB buscará todas las órdenes guardadas en la Base de Datos y las cargará al listado de la aplicación.

Entidades

Bicicleta

Esta es la clase abstracta y genérica base de la cual heredan *Mountain Bike* y *Playera*. Recibe un parámetro de clase de tipo Material, el cual será utilizado para inicializar un objeto de tipo Cuadro correspondiente a la bicicleta que se esté instanciando.

Implementa la interfaz *IBicicleta*, que define métodos para acceder a los campos de una Bicicleta sin especificar el parámetro de la clase (Material).

MountainBike

Hereda de Bicicleta, define un campo adicional *TieneCambios*. En base a este campo, su costo puede ser mayor o menor.

Playera

Hereda de Bicicleta, define un campo adicional *FrenoContraPedal*. En base a este campo, su costo puede ser mayor o menor.

Material

Esta es la clase abstracta que define los campos de cada material (nombre y costo). Los materiales que hereden de esta clase deben implementar un constructor vacío para ser utilizados en la fabricación de una Bicicleta, debido a las restricciones que le impone a su tipo genérico.

Las clases derivadas de *Material* deben llamar al constructor base, pasando por parámetros el nombre del material y el costo.

Los costos de los materiales se obtienen desde la tabla Costos en la Base de Datos.

Accesorio

Esta es la clase abstracta de la cual heredan Casco y Luz. Define los campos esenciales (nombre y costo).

Las clases derivadas incluyen nuevos campos, en base a los cuales podrán modificar el costo del accesorio.

Cuadro

Esta clase define un componente de Bicicleta. El cuadro debe recibir una instancia de Material en su constructor, la cual se obtiene a partir del parámetro genérico de la clase Bicicleta (y sus derivadas).

Rueda

Esta clase define un componente de Bicicleta. Recibe el tamaño del rodado en su constructor y en base al mismo, establece el costo de la rueda.

Orden

Esta clase define una orden que se enviará a producción. Contiene 2 objetos que implemente *Ibicicleta* y 2 que implemente *IFabricable*, que representan a una MountainBike, una Playera, un Casco y una Luz, respectivamente.

ListOrden

Esta clase implementa un listado de órdenes, añadiéndole funcionalidad para el guardado a archivos.

Fábrica

Esta clase estática será accesible por el proyecto de formulario. La misma tendrá un *ListOrden*, así como también controlará el stock de materiales disponible.

Esta clase envuelve los métodos de AgregarOrden, ImprimirArchivos, correspondientes a *Orden* y *ListOrden*, para que los mismos sean accedidos a través de Fábrica y no directamente desde sus clases.

Define también los métodos para importar y exportar el listado de órdenes a binario.

Estructura de la Base de Datos

Se definió una base de datos con 7 tablas.

La tabla Stock guarda el stock de cada material disponible. Costos guarda los costos base definidos para la aplicación.

MountainBikes, Playeras, Cascos, Luces guardan cada componente de una orden, y sus registros referencian al id de una orden.

La tabla Ordenes guarda las órdenes. Cada orden solo guarda el costo y 4 banderas para indicar a la aplicación si se debe buscar en las tablas de MountainBike, Playera, Casco o Luz.

En la carpeta raíz del proyecto (donde está ubicado el archivo .sln) se encuentran dos archivos: uno con el nombre de “TP4_Henestroza_Alejandro.bak”, el cual es un backup de la Base de Datos, para que pueda ser regenerada.

En caso de que el archivo .bak no funcione, se incluyó un segundo archivo “script.sql”, que ***en teoría*** recrea la base de datos, las tablas y los datos ingresados en las tablas.

Dónde encontrar cada tema visto en la cursada

Excepciones:

En el proyecto Entidades están definidas las Excepciones utilizadas. Las mismas se lanzan en varios momentos de la aplicación. StockInsuficienteException se lanza al intentar crear una orden sin contar con el stock suficiente. CamposInvalidosException se lanza si se intenta crear una orden sin completar todos los campos requeridos. RuedaIncompatibleException no se lanza nunca, dado que ya no se permite introducir cualquier valor en los ComboBox, pero solía lanzarse si se trataba de introducir un rodado inválido o si se intentaba crear una bicicleta con dos ruedas distintas.

Interfaces:

En el proyecto Interfaces se definen dos interfaces: IFabricable y IBicicleta.

IFabricable representa a cualquier producto fabricable (bicicletas y accesorios) y permite recuperar el Costo del producto. IBicicleta hereda de IFabricable, definiendo métodos adicionales para recuperar los campos de una bicicleta, sin tener que instanciar una (dado que es una clase genérica, de esta forma no es necesario especificar el material de la bicicleta).

Tipos Genéricos:

La clase Bicicleta (y sus derivadas) utiliza tipos genéricos para definir el material del cuadro de la Bicicleta. Se han aplicado dos restricciones: el tipo debe ser Material (o, al ser una clase abstracta, una de sus derivadas) y debe implementar un constructor sin parámetros. Por esta razón, todos los materiales deben implementar un constructor vacío, que llame al constructor de la clase base Material.

Archivos:

La clase ListOrden y la clase estática Fábrica implementas los dos temas vistos: Streams y Serialización.

Fábrica se encarga de Serializar el ListOrden a binario, mientras que ListOrden puede exportar cada Orden contenida a un archivo de texto.

Eventos y delegados:

La clase Orden define un evento OrdenCreada, del tipo DelGuardarOrden. Este evento se lanza tras crear una orden correctamente y al querer guardarla en la Base de Datos. En el formulario frmOrden se definen 4 métodos manejadores, uno por cada producto que pueda incluir la orden. Al agregar dicho producto a la orden, se suscribe el método manejador que se encarga de guardar el producto en la Base de Datos.

Hilos:

No encontré muchas formas de utilizar hilos en el proyecto, por lo que decidí utilizarlos para las conexiones a la Base de Datos. Cada vez que se trata de insertar un dato, actualizar uno o borrar uno de la Base de Datos, para no bloquear el flujo de la aplicación, se instancia un nuevo hilo con la funcionalidad indicada.